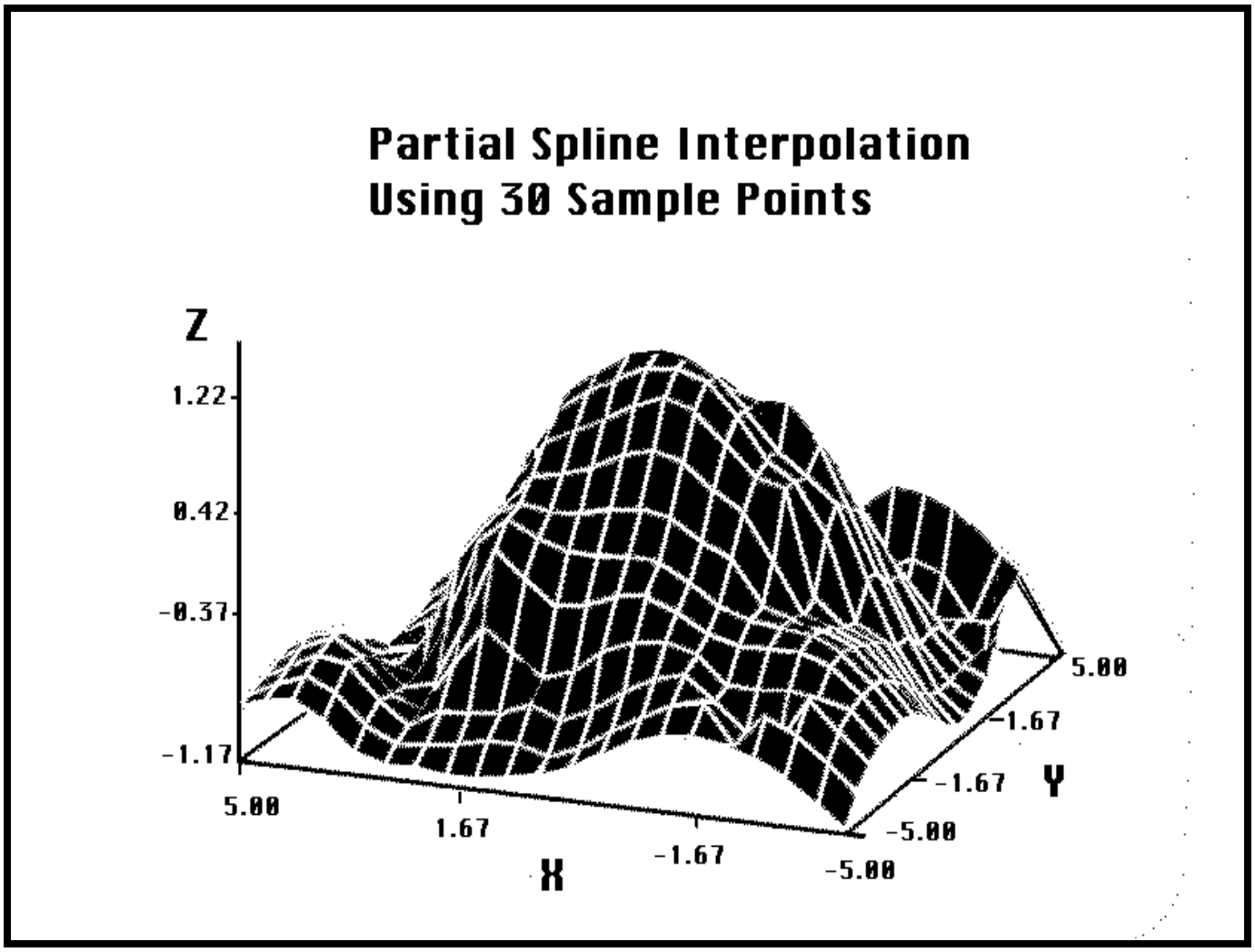
SAS
PRACTICAL USER'S GUIDE



This guide was developed by,
Samia Massoud, Ph. D.
Padmanabh M. Padaki
Charlie Apter, Graduate Assistant
Derya Guven, Graduate Assistant

©1993 Computing Services Center

TABLE OF CONTENTS

Introduction 1
SAS System Overview 1
What is SAS? 1
A SAS Job 1
Summary 2
How to Input Data into a SAS Program 2
Data Statement 2
Input Statement 3
Skipping Data 3
Long INPUT Statements 3
Cards Statement 3
Infile Statement 3
List Statement 4
Data Manipulation 4
IF Statements 4
Creating New Variables 5
Summary 5
SAS Procedures 5
Proc Print 6
Proc Freq 7
Summary 9
References 9

Appendix A: A List of Frequently-Used SAS Procedures 10
SAS Procedures for Statistical Analysis 10
Procedures for Handling SAS Libraries and Data Sets 10
Procedures for Manipulating Variables within SAS Data Sets 10
Procedures for Manipulating SAS Output 10
Appendix B: Some Examples of Job Control Language (JCL) for Running SAS 12
Appendix C: Some More Complicated Examples 15
Appendix D: SAS on VM/CMS 18

INTRODUCTION

This handout covers the essentials of inputting data into the SAS system, as well as some of the more commonly-used basic SAS commands. While SAS statements are independent of the computer system being used, the initial part of this manual (especially the parts regarding JCL) is oriented to the WYLBUR user. Appendices can be found in the bac; the user must therefore obtain specific information about the operating system of their choice from other materials. This handout was designed to provide the user with a very basic understanding of SAS and was not intended to exhaustively document the system. Many manuals already exist that document the many features available in SAS; some of these are listed in the reference section. The first section of this manual gives a brief overview of the SAS system and defines some of the terms and concepts used throughout the handout. The second section discusses how to get data into the system for processing, and how to manipulate this inputted data to produce a desired form. The final section explains several of the more important procedure statements and their uses. Appendix A lists many of the procedures statements, Appendix B includes examples of JCL for running a variety of SAS jobs, and Appendix C contains a few more complicated examples.

SAS SYSTEM OVERVIEW

WHAT IS SAS?

SAS is a software system for data analysis. This means that SAS is a computer program that takes data provided by the user and statistically analyzes it, checking for errors, performing chosen procedures, and printing the results, as requested by the user.

A SAS JOB

A SAS job, divided into 4 sections as shown below, is a set of SAS statements assembled to perform data analysis and produce desired output.

PART 1: JCL

```
//JOBNAME JOB (,box,time,lines),'comment',USER=logon-id
//STEP EXEC SAS
//SYSIN DD *
```

PART 2: DATA STEP

```
DATA EXAMPLE;
INPUT NAME $ 1-15 SEX $ 16 AGE 18-19 GPR 21-23;
LIST;
CARDS;
```

PART 3: DATA

```
ADAMS          M 20 2.6
BAKER          F 19 3.2
DOUGLAS        M 19 2.8
HALE           M 21 4.0
JONES          F 19 2.5
LARUE          F 18 3.8
NICKS          F 19 2.9
OLAJUWON       M 22 1.8
PEBBLES        F 22 2.4
RAINES         F 21 2.8
SMITH          M 20 2.7
TAYLOR         F 19 3.4
```

PART 4: PROCEDURES

```
PROC PRINT;
```

A typical SAS program (like the example above) consists of four major sections:

- **Job control language (JCL)**. These commands tell the computer who is using it and what program is being executed (in this case, SAS). The commands listed here are those for running SAS on the MVS (WYLBUR) system, which is most widely used at Texas A&M. A listing of other MVS system commands commonly used to run SAS are found in Appendix B. To run SAS on other operating systems, the user should consult the Help Manual for that particular system.

- **Data step.** This is usually where the user first begins inputting SAS statements. In this section, the data variable names are defined and the data is assigned to each variable. Any data conversions that may need to be done are carried out in this step. Data input and manipulation are discussed in Chapter 2. You may have noticed that all of the statements in this section end with a semicolon (;) -- this is a requirement for ALL SAS statements found in the Data step.

- **The data.** The third section of our example is the data that the user wants to analyze. If the user is keying the data in, as in the example, it should appear at this point in the program. As you will learn later, it will not appear here if the data is in a separate file. Note that only one semicolon (;) appears in the Data, at the very end of the section.

- **Procedure (PROC) statements.** The final section contains SAS procedure statements that describe analyses to be performed. Some of the many SAS statements that are available are discussed in the third section (**SAS Procedures**). Again, every statement ends with a semicolon (;).

SUMMARY

The following is a summary of the steps to follow using SAS:

1. Collect the data and assemble it in a form the computer can read.
2. Put together the SAS job.
3. Submit the job to the computer and get the printed results.

HOW TO INPUT DATA INTO A SAS PROGRAM

The first thing you need to do is get your data into a form that the computer can read. This requires up to four different SAS statements:

- **DATA**
- **INPUT**
- **INFILE**
- **CARDS**

This section introduces these four statements and ties them together.

DATA

The `DATA` statement is usually the first statement in a SAS job; it begins with the word `DATA` and is followed by a name that you choose for the data set. Data set names must begin with a letter, and can be no more than 8 characters in length. The form for the `DATA` statement is `DATA dsname;`

INPUT

Each line of data in a SAS program can be an observation; each value in this observation represents a variable, and the INPUT statement is used to name these variables. The INPUT statement follows the DATA statement. For example, to describe the following line of data,

```
ADAMS      M 20 2.6
```

you would begin with the word INPUT followed by the name of |the first variable, which is NAME. Since this variable is non-numeric, a dollar sign ($) must be placed after it. This is done only in the INPUT statement. In all subsequent uses in the job, the $ is omitted. In this example, the name ADAMS begins in column 1 and ends in column 5. However, other names in the data set may be longer, so room must be allocated for them. 15 spaces would probably be enough. Skip a space after the variable name and put the first and last column numbers, separated by a dash ('-'). Repeat this for each variable. The INPUT statement for the above example would be:

```
INPUT NAME $ 1-15 SEX $ 16 AGE 18-19 GPR 21-23;
```

Please note that it is not mandatory that column numbers be specified; as long as one blank space is inserted between each variable value in an observation, SAS will read them as separate values. Some special situations involving the INPUT statement exist that one should be aware of. They include:

- **Skipping Data.** You may not want to use all the variables in a data set. By omitting the variable name (and the corresponding column numbers) in the INPUT statement, SAS will not include its values in any computations. Be sure that the variables you DO want (and their column numbers) are included and appear correctly.

- **Long INPUT Statements.** If you have an INPUT statement that exceeds the length of one line, simply continue it on the next line; be sure that variable names are not broken between lines, and that a semicolon only appears at the END of the statement. For example:

```
INPUT NAME $ 1-15 SEX $ 16 AGE 18-19
GPR 21-23;
```

CARDS

When data is entered as an internal part of a SAS program, the CARDS statement immediately precedes the data lines. It is simply entered as CARDS; and tells the SAS system that the data follows. Note that when a CARDS statement is used, the line length cannot exceed 80 columns (characters).

INFILE

Data may also be 'imported' from a disk or tape into your SAS program. In this case, both the computer operating system and SAS must know where the data is to be found. An INFILE statement is used to accomplish this. The INFILE statement goes before the INPUT statement. It consists of INFILE followed by the file reference name. This identifies the name of the file to be used. For example, if you are using a file called STUDENTS, the statement would be

```
INFILE STUDENTS;
```

Please note that the CARDS and INFILE statements are not used together. Using our example from before, but with an INFILE statement, it looks like this:

```
DATA EXAMPLE;
   INFILE 'ABC1234.STUDENTS';
   INPUT NAME $ 1-15 SEX $ 16 AGE 18-19 GPR 21-23;
SAS statements
```

Indenting the `INFILE` and `INPUT` statements is not necessary. It does help, however, when reading the program. When using the `INFILE` statement, you must tell the computer's operating system where the data can be located. This is done immediately following the `DATA` statement, as shown in Appendix B. The system commands (JCL) for reading external files into the WYLBUR system are listed in Appendix B.

LIST

The `LIST` statement is an optional statement. It goes after the `INPUT` statement. The purpose of the `LIST` statement is to list each line of data as it is read in. It is useful for editing and debugging. It allows you to see if all the data you wanted was read in and if it was read correctly. The `LIST` statement should appear in the form `LIST;`

DATA MANIPULATION

Very often data is not in its desired form, or calculations based on the data are desired. SAS allows you to do this by using what are called program statements. Available program statements include arithmetic operations, `IF` statements, comparison operators, and others that are beyond the scope of this handout. The tables below list the arithmetic and comparison operators available and their SAS equivalent.

Arithmetic Operators		Comparison Operators		
**	Exponential	<	LT	Less than
*	Multiplication	<=	LE	Less than or equal
/	Division	>	GT	Greater than
+	Addition	>=	GE	Greater than or equal
-	Subtraction	=	EQ	Equal
		~=	NE	Not equal

All program statements go after the `INPUT` statement, but before the `CARDS` statement (if there is one). Using these operators allows you to create new variables, and with the `IF` statement, allow one to 'convert' data into another form. The next two sections explain `IF` statements and creation of new variables.

● **IF statements.** With the `IF` statement, one can control what portion of the data is processed. The form for the `IF` statement is `IF condition THEN statement ELSE statement;`
Using the data from the previous examples, to only process data on females, the following commands would be used:

```
DATA EXAMPLE;
INPUT NAME $ 1-15 SEX $ 16 AGE 18-19 GPR 21-23;
   IF SEX = 'F';
CARDS;
```

In this example the THEN and ELSE were not necessary. They are optional. Since F is alphabetic, it must appear in quotes. If, for example, you only want to process students that are over 20, you would use the following:

```
DATA EXAMPLE;
INPUT NAME $ 1-15 SEX $ 16 AGE 18-19 GPR 21-23;
   IF AGE GE 21;
CARDS;
```

You can also use AND's and OR's in your comparisons. Such as:

```
DATA EXAMPLE;
INPUT NAME $ 1-15 SEX $ 16 AGE 18-19 GPR 21-23;
   IF AGE LT 20 OR AGE GT 20;
CARDS;
```

This would eliminate all observations with an age of 20.

● **Creating new variables.** One may want to create new variables that do not appear in the input data set. For example, in the data set above, one may wish to create a status variable that identifies students with high GPR's (GT 3.2) or low GPR's (LT 2.0). The variable STATUS could be created such that it would have a value of 1 for high GPR's, 3 for low GPR's, and 2 for all other GPR's, as below:

```
DATA EXAMPLE;
INPUT NAME $ 1-15 SEX $ 16 AGE 18-19 GPR 21-23;
   IF GPR GT 3.2 THEN STATUS=1;
   IF GPR LT 2.0 THEN STATUS=3;
   IF GPR GE 2.0 AND GPR LE 3.2 THEN STATUS=2;
CARDS;
```

SUMMARY

Some important things to remember are: to input data into SAS, a DATA statement must be used, followed by an INFILE statement, or by INPUT and CARDS statements. Data manipulation is only limited by your imagination, as long as you use the available program statements and follow their format. This section contained only a couple of examples. Check available manuals if you are interested in others.

SAS PROCEDURES

Procedure statements (PROCs) are used to analyze, summarize, etc. the data once it has been added to a SAS job. Many PROC statements exist in SAS. All begin with PROC followed by the procedure to be performed. Only a few of the many PROC statements will be introduced in this section; a description of several other procedures can be found in Appendix A.

Each PROC statement executes some operation on the data and prints the results. They begin immediately following the lines of data, if using the CARDS statement, or following the INPUT statement if using an INFILE statement. This handout discusses the use of PROC PRINT and PROC FREQ (frequency), two of the more commonly used statements.

PROC PRINT

Under most circumstances, you will want a printout of your data. This is done with the PRINT procedure. The data is printed in columns labeled with their name (i.e., columns are labelled with the variable name). For example, the following SAS job

```
DATA EXAMPLE;
INPUT NAME $ 1-15 SEX $ 16 AGE 18-19 GPR 21-23;
CARDS;
ADAMS     M 20 2.6
BAKER     F 19 3.2
DOUGLAS   M 19 2.8
HALE      M 21 4.0
JONES     F 19 2.5
LARUE     F 18 3.8
NICKS     F 19 2.9
OLAJUWON  M 22 1.8
PEBBLES   F 22 2.4
RAINES    F 21 2.8
SMITH     M 20 2.7
TAYLOR    F 19 3.4
;
PROC PRINT;
```

would result in the following output:

```
                    The SAS System

OBS   NAME          SEX   AGE   GPR

1     ADAMS         M     20    2.6
2     BAKER         F     19    3.2
3     DOUGLAS       M     19    2.8
4     HALE          M     21    4.0
5     JONES         F     19    2.5
6     LARUE         F     18    3.8
7     NICKS         F     19    2.9
8     OLAJUWON      M     22    1.8
9     PEBBLES       F     22    2.4
10    RAINES        F     21    2.8
11    SMITH         M     20    2.7
12    TAYLOR        F     19    3.4
```

If you want the variables in a certain order and/or only some of the variables printed, you would use the variable statement. This is done by following the PROC PRINT statement with VAR (for variable) and the names of the variables in the order desired. In the example above, if you only wanted the NAME and GPR, the following would be used:

```
PROC PRINT;
  VAR NAME GPR;
```

The resulting output would be:

The SAS System

OBS	NAME	GPR
1	ADAMS	2.6
2	BAKER	3.2
3	DOUGLAS	2.8
4	HALE	4.0
5	JONES	2.5
6	LARUE	3.8
7	NICKS	2.9
8	OLAJUWON	1.8
9	PEBBLES	2.4
10	RAINES	2.8
11	SMITH	2.7
12	TAYLOR	3.4

The format and content of output can be controlled in more detail with other statements. Consult the **SAS Procedures Guide** cited in the reference section for more information.

PROC FREQ

Often a summary of the data is desired; frequency tables are one way in SAS to accomplish this. The format for the frequency table is:

```
PROC FREQ;
  TABLES variables;
```

Follow tables with the variable you wish to summarize. The output will contain the frequency of all the different values for that variable. Using the example data set you may want to know the frequency of males and females. You would get this with the following statements:

```
PROC FREQ;
  TABLES SEX;
```

And get these results:

The SAS System

SEX	FREQUENCY	PERCENT	CUMULATIVE FREQUENCY	CUMULATIVE PERCENT
F	7	53.8	7	53.8
M	6	46.2	13	100.0

There may also be times when you want to break the variables down even further. For example by age and sex. This is called crosstabulation and it gives you a two dimensional table like this:

The SAS System

TABLE OF AGE BY SEX

AGE / FREQUENCY / PERCENT / ROW PCT / COL PCT	SEX F	SEX M	TOTAL
18	1	0	1
	7.69	0.00	7.69
	100.00	0.00	
	14.29	0.00	
19	4	1	5
	30.77	7.69	38.46
	80.00	20.00	
	57.14	16.67	
20	0	2	2
	0.00	15.38	15.38
	0.00	100.00	
	0.00	33.33	
21	1	1	2
	7.69	7.69	15.38
	50.00	50.00	
	14.29	16.67	
22	1	2	3
	7.69	15.38	23.08
	33.33	66.67	
	14.29	33.33	
TOTAL	7	6	13
	53.85	46.15	100.00

The only change in the statement is to add the next variable and separate it from the first by a asterisk (*). For the above table the statement would be:

```
PROC FREQ;
  TABLES AGE*SEX;
```

To do a 3 way crosstabulated frequency table, we could do the following:

```
PROC FREQ;
  TABLES STATUS*SEX*AGE;
```

And the output:

TABLE 1 OF SEX BY AGE
CONTROLLING FOR STATUS=1

SEX / AGE / FREQUENCY / PERCENT / ROW PCT / COL PCT	18	19	20	21	22	TOTAL
F	1	1	0	0	0	2
	25.00	25.00	0.00	0.00	0.00	50.00
	50.00	50.00	0.00	0.00	0.00	
	100.0	100.0	.	0.00	0.00	
M	0	0	0	1	1	2
	0.00	0.00	0.00	25.00	25.00	50.00
	0.00	0.00	0.00	50.00	50.00	
	0.00	0.00	.	100.0	100.0	
TOTAL	1	1	0	1	1	4
	25.00	25.00	0.00	25.00	25.00	100.00

TABLE 2 OF SEX BY AGE
CONTROLLING FOR STATUS=2

SEX FREQUENCY PERCENT ROW PCT COL PCT	AGE 18	19	20	21	22	TOTAL
F	0 0.00 0.00 75.00	3 37.50 60.00 0.00	0 0.00 0.00 100.0	1 12.50 20.00 100.0	1 12.50 20.00	5 62.50
M	0 0.00 0.00 25.00	1 12.50 33.33 100.0	2 25.00 66.67 0.00	0 0.00 0.00 0.00	0 0.00 0.00	3 37.50
TOTAL	0 0.00	4 50.00	2 25.00	1 12.50	1 12.50	8 100.00

TABLE 3 OF SEX BY AGE
CONTROLLING FOR STATUS=3

SEX FREQUENCY PERCENT ROW PCT COL PCT	AGE 18	19	20	21	22	TOTAL
F	0 0.00 . .	0 0.00 . .	0 0.00 . .	0 0.00 . .	0 0.00 . 0.00	0 0.00
M	0 0.00 0.00 .	0 0.00 0.00 .	0 0.00 0.00 .	0 0.00 0.00 .	1 100.00 100.00 100.00	1 100.0
TOTAL	0 0.00	0 0.00	0 0.00	0 0.00	1 100.00	1 100.00

The first variable in the TABLES statement (STATUS) is divided into the three tables above. The second variable (SEX) is used as the rows in each table. The third variable (AGE) is used as the columns in each table.

SUMMARY

PROC PRINT and PROC FREQ are just two of the many procedures available in SAS. There are also many options which can be added to these procedures for more detailed analysis. The SAS manuals listed in the Reference section give detailed descriptions of many of the procedures and utilities available on the SAS system.

REFERENCES

SAS Language: Reference. Version 6, First Edition. 1990.SAS Institute, Inc., Cary, NC.

SAS Procedures Guide. Version 6, Third Edition. 1990. SAS Institute, Inc., Cary, NC.

SAS Language and Procedures: Usage. Version 6, First Edition. 1989. SAS Institute Inc., Cary, NC.

SAS/STAT User's Guide, Volume 1 and 2. Version 6, Fourth Edition. 1990. SAS Institute Inc., Cary, NC.

APPENDIX A

A List of Frequently-Used SAS Procedures

SAS PROCEDURES FOR STATISTICAL ANALYSIS

PROC ANOVA - Performs analysis of variance for balanced data.

PROC CORR - Computes correlation coefficients between variables.

PROC FREQ - Produces one-way and n-way frequency and crosstabulation tables.

PROC GLM - Uses the method of least-squares to form general linear models. Can be used for regression, analysis of variance, analysis of covariance, multivariate ANOVA, and partial correlation.

PROC MEANS - Produces simple univariate descriptive statistics for numeric variables.

PROC NLIN - Produces least-squares or weighted least-squares estimates of the parameters of a nonlinear model.

PROC REG - Fits least-squares estimates to linear regression models.

PROC UNIVARIATE - Produces simple descriptive statistics for numeric variables.

PROCEDURES FOR HANDLING SAS LIBRARIES AND DATA SETS

PROC CONTENTS - Prints descriptions of the contents of one or more files from a SAS library.

PROC CONVERT - Converts BMDP, DATA-TEXT, OSIRIS, and SPSS files to SAS data sets.

PROC COPY - Copies an entire SAS data library or selected members of the library.

PROC DATASETS - Used to modify members within a SAS data library.

PROC PDS - Can list, delete, and rename the members of a partitioned data set.

PROC PDSCOPY - Copies partitioned data sets containing load modules between storage devices (tapes and disks).

PROC RELEASE - Releases unused space at the end of a disk data set.

PROC SOURCE - Provides an easy way to back up and process library data sets.

PROC TAPECOPY - Copies an entire tape volume or files from one or several tape volumes to one output tape volume.

PROC TAPELABEL - Lists the label information of an IBM standard labeled tape volume.

PROCEDURES FOR MANIPULATING VARIABLES WITHIN SAS DATA SETS

PROC APPEND - Adds the observations from one SAS data set to the end of another SAS data set.

PROC SORT - Sorts observations in a SAS data set by one or more variables.

PROC TRANSPOSE - Transposes a SAS data set, changing observations into variables and vice versa.

PROCEDURES FOR MANIPULATING SAS OUTPUT

PROC CALENDAR - Displays data from a SAS data set in a month-by-month calendar format.

PROC CHART - Produces bar charts, block charts, pie charts, and star charts.

PROC FORMAT - Used to define the output format for character and numeric values.

PROC PLOT - Graphs one variable against another, producing a printer plot.

PROC PRINT - Prints the observations in a SAS data set, using all or some of the variables.

PROC PRINTTO - Used to define the destination for SAS procedure output.

PROC SUMMARY - Computes descriptive statistics on numeric variables and outputs the results to a new SAS data set.

PROC TABULATE - Constructs tables of descriptive statistics from compositions of classifi-cation variables, analysis variables, and statistics keywords.

APPENDIX B

Some Examples of Job Control Language (JCL) for Running SAS

NOTE: In the following examples, a generic JOB 'card' was used. If, for example, your logon-id (account number or DPSR number) is `ABC1234`, and you want to allow `14` seconds for CPU time and produce no more than `5000` lines of output which you would find in box `5A`, then your job card should read as follows:

```
//jobname JOB (,5A,S14,5),'MYNAME',USER=ABC1234
```

EXAMPLE 1: Data in the job stream.

```
//jobname JOB (,box,time,lines),'comment',USER=logon-id
//*TAMU HOLDOUT,NOTIFY,PRTY=4
//STEP EXEC SAS
//SYSIN DD *
;
DATA ONE;
INPUT statement;
other SAS statements;
CARDS;
Insert data here.  Each line of data must be no more than 80 columns wide.
```

EXAMPLE 2: Reading data from a cataloged external 'flat file' (ASCII). NOTE: `RAWDATA` file should be in fixed block format.

```
//jobname JOB (,box,time,lines),'comment',USER=logon-id
//*TAMU HOLDOUT,NOTIFY,PRTY=4
//STEP EXEC SAS
//SYSIN DD *
;
DATA ONE;
INFILE 'ABC1234.RAWDATA';
INPUT statement;
other SAS statements;
```

EXAMPLE 3: Reading data from a cataloged external SAS dataset.

```
//jobname JOB (,box,time,lines),'comment',USER=logon-id
//*TAMU HOLDOUT,NOTIFY,PRTY=4
//STEP EXEC SAS
//SYSIN DD *
;
LIBNAME IN 'ABC1234.SASDATA';
DATA XYZ;
SET IN.SASDATA;
other SAS statements;
```

EXAMPLE 4: Reading data from a cataloged external 'flat file' (ASCII) and creating another cataloged 'flat file' on disk. `RAWDATA` file should be in fixed block format; `BLKSIZE b` (<6356) should be a multiple of `l` (<=232).

```
//jobname JOB (,box,time,lines),'comment',USER=logon-id
//*TAMU HOLDOUT,NOTIFY,PRTY=4
//STEP EXEC SAS
//SYSIN DD *
;
FILENAME OUT 'ABC1234.NEWFILE' DISP=(NEW,CATLG,DELETE)
   UNIT=DISK SPACE=(TRK,(10,5),RLSE)
   LRECL=l RECFM=FB BLKSIZE=b;
;
DATA ONE;
   INFILE 'ABC1234.RAWDATA';
   INPUT statement;
   other SAS statements;
   FILE OUT;
   PUT statement;
```

EXAMPLE 5: Reading data from a cataloged external 'flat file' (ASCII) and creating a cataloged SAS dataset on disk.

```
//jobname JOB (,box,time,lines),'comment',USER=logon-id
//*TAMU HOLDOUT,NOTIFY,PRTY=4
//STEP EXEC SAS
//SYSIN DD *
;
LIBNAME OUT 'ABC1234.NEWFILE' DISP=(NEW,CATLG,DELETE)
   UNIT=DISK SPACE=(TRK,(10,5),RLSE);
;
DATA OUT.NEWFILE;
   INFILE 'ABC1234.RAWDATA';
   INPUT statement;
   other SAS statements;
```

EXAMPLE 6: Reading data from an uncataloged external 'flat file' on a non-labeled tape.

```
//jobname JOB (,box,time,lines),'comment',USER=logon-id
//*TAMU HOLDOUT,NOTIFY,PRTY=4
//STEP EXEC SAS
//IN DD DISP=SHR,UNIT=TAPE9,
//      VOL=SER=TAPE#,LABEL=(n,NL,,IN),
//      DCB=(RECFM=FB,LRECL=l,BLKSIZE=b)
//* note: code the right DCB parameters.
//*       n is the file number to be read.
//* note: l and b MUST be integers and B must be less than 32760.
//SYSIN DD *
;
DATA ONE;
   INFILE IN options; /* refer to Language Guide for more on options */
   INPUT statement;
   other SAS statements;
```

EXAMPLE 7: Reading data from an uncataloged external 'flat file' on a standard label 9-track tape.

```
//jobname JOB (,box,time,lines),'comment',USER=logon-id
//*TAMU HOLDOUT,NOTIFY,PRTY=4
//STEP EXEC SAS
//IN DD DISP=SHR,UNIT=TAPE9,DSN=file1,
//    VOL=SER=TAPEX,LABEL=(n,SL,,IN)
//*   FOR CARTRIDGES UNIT=TAPEC
//SYSIN DD *
;
DATA ONE;
   INFILE IN options;
   INPUT statement;
   OTHER SAS statements;
```

APPENDIX C
Some More Complicated Examples

The command `PROC PRINT;` will give you a printout of your data. For example, the following SAS job:

```
DATA EXAMPLE;
INPUT CASE PROD WIDTH DENS STR ID$;
CARDS;
1 763 19.8 128 86 A
2 650 20.9 110 72 B
3 554 15.1 95 62 C
4 742 19.8 123 82 D
5 470 21.4 77 52 E
6 651 19.5 107 72 F
7 756 25.2 123 84 G
9 681 26.8 116 76 I
10 579 28.8 100 64 J
11 716 22.0 110 80 K
12 650 24.2 107 71 L
13 761 24.9 125 81 M
14 549 25.6 89 61 N
15 641 24.7 103 71 O
16 606 26.2 103 67 P
17 696 21.0 110 77 R
18 795 29.4 133 83 S
19 582 21.6 96 65 T
20 559 20.0 91 62 U
;
PROC PRINT;
```

would result in the following output:

OBS	CASE	PROD	WIDTH	DENS	STR	ID
1	1	763	19.8	128	86	A
2	2	650	20.9	110	72	B
3	3	554	15.1	95	62	C
4	4	742	19.8	123	82	D
5	5	470	21.4	77	52	E
6	6	651	19.5	107	72	F
7	7	756	25.2	123	84	G
8	9	681	26.8	116	76	I
9	10	579	28.8	100	64	J
10	11	716	22.0	110	80	K
11	12	650	24.2	107	71	L
12	13	761	24.9	125	81	M
13	14	549	25.6	89	61	N
14	15	641	24.7	103	71	O
15	16	606	26.2	103	67	P
16	17	696	21.0	110	77	R
17	18	795	29.4	133	83	S
18	19	582	21.6	96	65	T
19	20	559	20.0	91	62	U

Proc corr is the one way to show the correlation between variables. For example,

```
PROC CORR;
   VAR PROD WIDTH DENS STR;
```

results in following output:

SIMPLE STATISTICS

VARIABLE	N	MEAN	STD DEV	SUM	MINIMUM	MAXIMUM
PROD	19	652.684	89.719	2401.0	470.0	795.0
WIDTH	19	22.995	3.628	436.9	15.1	29.4
DENS	19	107.684	14.678	2046.0	77.0	133.0
STR	19	72.000	9.452	1368.0	52.0	86.0

PEARSON CORRELATION COEFFICIENTS / PROB > |R| UNDER HO: RHO=0 / N = 19

	PROD	WIDTH	DENS	STR
PROD	1.00000	0.21452	0.97754	0.98952
	0.00000	0.37780	0.00010	0.00010
WIDTH	0.21452	1.00000	0.24577	0.14777
	0.37780	0.00000	0.31050	0.54600
DENS	0.97754	0.24577	1.00000	0.96268
	0.00010	0.31050	0.00000	0.00010
STR	0.98952	0.14777	0.96268	1.00000
	0.00010	0.54600	0.00010	0.00000

The regression (REG) procedure fits linear regression models using the least-squares procedure; for example, the statements

```
PROC REG;
   MODEL PROD=WIDTH DENS STR/P R XPX I VIF COLLINOINT INFLUENCE PARTIAL;
   OUTPUT OUT=NEW;
```

results in the following output:

MODEL CROSSPRODUCTS X'X X'Y Y'Y

X'X	INTERCEP	WIDTH	DENS	STR	PROD
INTERCEP	19.0	436.9	2046.0	1368.0	12401.0
WIDTH	436.9	10283.2	47282.8	31548.0	286414.5
DENS	2046.0	47282.8	224200.0	149716.0	1358564.0
STR	1368.0	31548.0	149716.0	100104.0	907976.0
PROD	12401.0	286414.5	1358564.0	907976.0	8238829.0

X'X INVERSE, PARAMETER ESTIMATES, AND SSE

	INTERCEP	WIDTH	DENS	STR	PROD
INTERCEP	5.0886	-0.0959	0.0334	-0.0892	-42.2676
WIDTH	-0.0959	0.0051	-0.0018	0.0024	0.9862
DENS	0.0334	-0.0018	0.0041	-0.0061	1.7382
STR	-0.0892	0.0024	-0.0061	0.0096	6.7386
PROD	-42.2676	0.9826	1.7382	6.7386	1598.8798

Variable: PROD

ANALYSIS OF VARIANCE

SOURCE	DF	SUMS OF SQUARES	MEAN SQUARE	F VALUE	PROB>F
MODEL	3	143293.22543	47764.40848	448.105	0.0001
ERROR	15	1598.87983	106.59199		
C TOTAL	18	144892.10526			

ROOT MSE	10.32434	R-SQUARE	0.9890
DEP MEAN	652.68421	ADJ R-SQ	0.9868
C.V.	1.58183		

PARAMETER ESTIMATES

VARIABLE	DF	PARAMETER ESTIMATE	STANDARD ERROR	T FOR H0: PARAMETER=0	PROB>|T|	VARIANCE INFLATION
INTERCEP	1	-42.26760	23.2893832	-1.815	0.0896	0.0000000
WIDTH	1	0.98246	0.7354680	1.336	0.2015	1.2021227
DENS	1	1.73821	0.6642529	2.617	0.0194	16.0532136
STR	1	6.73863	1.0110315	6.665	0.0001	15.4202328

COLLINEARITY DIAGNOSTICS (INTERCEPT ADJUSTED)

NUMBER	EIGENVALUE	CONDITION INDEX	VAR PROP WIDTH	VAR PROP DENS	VAR PROP STR
1	2.03749	1.00000	0.0275	0.0145	0.0146
2	0.93036	1.47986	0.8289	0.0011	0.0039
3	0.03214	7.96154	0.1436	0.9843	0.9815

The factor procedure performs several types of common factor and component analysis. You can compute scoring coefficients by the regression method, and you can write estimated factor scores to an output data set.

```
PROC FACTOR SIMPLE CORR MINEIGEN=0 EV NFACTORS=3 OUT=SCORES;
  VAR WIDTH DENS STR;
```

The output resulting from these statements would be:

MEANS AND STANDARD DEVIATIONS FROM 19 OBSERVATIONS

	WIDTH	DENS	STR
MEAN	22.9947368	107.684211	72.0000000
STD DEV	3.6277439	14.678225	9.4516312

CORRELATIONS

	WIDTH	DENS	STR
WIDTH	1.00000	0.24577	0.14777
DENS	0.24577	1.00000	0.96268
STR	0.14777	0.96268	1.00000

PRIOR COMMUNALITY ESTIMATES: ONE
EIGENVALUES OF THE CORRELATION MATRIX: TOTAL=3 AVERAGE=1

	1	2	3
EIGENVALUE	2.0375	0.9304	0.0321
DIFFERENCE	1.1071	0.8982	
PROPORTION	0.6792	0.3101	0.0107
CUMULATIVE	0.6792	0.9893	1.0000

3 FACTORS WILL BE RETAINED BY THE NFACTOR CRITERION

EIGENVECTORS

	1	2	3
WIDTH	0.25961	0.96284	0.07449
DENS	0.68919	-0.13069	-0.71269
STR	0.67647	-0.23636	0.69751

FACTOR PATTERN

	FACTOR1	FACTOR2	FACTOR3
WIDTH	0.37057	0.92871	0.01335
DENS	0.98376	-0.12606	-0.12778
STR	0.96560	-0.22798	0.12505

APPENDIX D
SAS on VM/CMS

SAS on VM/CMS can be run in two modes: interactive or non-interactive. This section describes the latter. An advantage to running SAS on VM is that no job control language (JCL) is required (it is required for SAS jobs run on Wylbur).

Three types of files

If SAS statements are collected in a file named `SAMPLE`, the program name must be `SAMPLE SAS A`. Error messages and other notes generated by SAS will be stored in `SAMPLE LOG A`. Output from the `PROC`s are saved in `SAMPLE LISTING A`. NOTE that each time SAS is executed non-interactively, the `LOG` and the `LISTING` files are replaced with a new copy. To avoid this replacement, see the SAS Companion for the VM/CMS Operating System, Chapter 8.

Creating a SAS program

To create a SAS source program, use the XEDIT command. If you wish to create a SAS source program file named `SAMPLE`, the filetype must be `SAS`. For example, the statement `Xedit SAMPLE SAS` will create an empty file for you. You can then type your source program into the empty file. For information about using the **XEDIT** editor for creating SAS programs, see the **VM User's Guide**.

Accessing the SAS minidisk

Before you begin execution of SAS, you must first link to the SAS minidisk. The command for this is
`PRODUCTS ADD SAS`

Using the CARDS statement to indicate data

Input of embedded data in a program requires a `CARDS` statement; data will be in-stream with the program, and cannot be more than 80 characters long. For example:

```
DATA SAMPLE;
INPUT X Y;
SUM = X+Y;
CARDS;
0.33 1.25
;
PROC PRINT;
```

Reading an external file in SAS

`FILEDEF` statements are used to indicate external input or output files. See the **VM User's Guide** for more information regarding `FILEDEF` statements. **`INFILE` statements in SAS must accompany a `FILEDEF` statement**. If the program name is `SAMPLE SAS A`, output from `PROC`s are stored in a file named `SAMPLE LISTING A`. In this example, the data is stored in the file named `DATA`.

```
DATA SAMPLE;
INFILE 'INPUTF DATA';
INPUT X Y;
SUM = X+Y;
PROC PRINT;
```

Creating a SAS dataset.

SAS data sets can be created only by SAS DATA steps and SAS PROCEDURES. These data sets can only be analyzed using SAS statements. Creating a SAS dataset does not require the user to issue FILEDEF statements. Using the example above, we wish to create a SAS data set. This is done by using a two-part name in the DATA statement. In the example below, the statement DATA NEW.NUMBERS, defines NEW as the first level name, and NUMBERS as the SAS internal name. This name is placed in the SAS dataset library for later reference. The CMS filename is NUMBERS NEW A (notice that the order of names in the DATA statement is reversed.)

```
DATA NEW.NUMBERS;
INFILE 'INPUTF DATA';
INPUT X Y;
SUM = X+Y;
PROC PRINT DATA=NEW.NUMBERS;
```

Accessing a SAS dataset.

This is how you access the saved SAS dataset system file.

```
DATA NEW2;
SET NEW.NUMBERS;
PROC PRINT DATA=NEW2;
```

or

```
PROC PRINT DATA=NEW.NUMBERS;
```

Running a SAS program

To run a SAS program, enter the command SAS filename (options. Example:

```
SAS SAMPLE (options
```

where options controls such things as certain data set attributes, SAS output features, the efficiency of program execution, etc. For a listing of these options refer to the SAS Companion for the VM/CMS Operating System, Chapter 8.

Sample SAS Log

At the **Ready; T=0.01/0.01 12:33:45** prompt, type

```
PRODUCTS ADD SAS
```

When the **Ready; T=0.07/0.08 12:34:00** prompt is returned, the following program can be created by typing

```
X SAMPLE SAS A
```

Once this empty file is created, enter the following information in it:

```
DATA SAMPLE;
CMS FILEDEF RAWIN DISK INPUTF DATA A;
INFILE RAWIN;
INPUT X Y;
SUM = X + Y;
CMS FILEDEF FT12F001 TERM;
PROC PRINT;
```

When your entry of the above information is complete, type at the command line FILE, and when the **Ready; T=0.01/0.02 12:34:05** prompt is returned, type

```
SAS SAMPLE
```

When the **Ready; T=0.01/0.02 12:34:05** prompt is returned, type

```
FILEL
```

In your listing of files, the following should appear (along with other files that might be there):

```
SAMPLE SASLOG
SAMPLE LISTING
SAMPLE SAS
```

SAMPLE SAS is the original file you created; SAMPLE SASLOG contains a listing of the procedures you executed and any information SAS might provide about the execution of those procedures. SAMPLE LISTING includes the results of the PROC statements, including statistical analyses, etc. Using our example from above, the SAS LISTING would be:

```
                    SAS
OBS      X        Y       SUM
 1     0.33     1.25     1.58
```